Pattern Recognition, Vol. 23, No. 5, pp. 485–495, 1990
Printed in Great Britain

0031-3203/90 $3.00 + .00
Pergamon Press plc
© 1990 Pattern Recognition Society

ARABIC CHARACTER RECOGNITION SYSTEM: A STATISTICAL APPROACH FOR RECOGNIZING CURSIVE TYPEWRITTEN TEXT

SHERIF SAMI EL-DABI, REFAT RAMSIS and ALADIN KAMEL*
IBM Kuwait Scientific Center, P.O. Box 4175, Safat, Kuwait

(*Received* 4 *March* 1988; *in revised form* 28 *March* 1989; *received for publication* 6 *April* 1989)

Abstract—Character recognition systems can contribute tremendously to the advancement of automation process, and can improve the interface between man and machine (computers) in many applications, including office automation and data entry. In this report we present a recognition system for typed Arabic text, which involves a statistical approach for character recognition. This approach uses "Accumulative Invariant Moments" as an identifier, which helped in the segmentation of connected and overlapping Arabic characters. However, Invariant Moments proved to be very sensitive to slight changes in a character shape. These changes are normally due to typing and the scanning process, and cannot be avoided. The recognition zone was defined based on the mean and standard deviation for the moments of a large sample of each character. However, this zone was increased, using an empirical multiplier, to improve recognition rate. The system was implemented on a mainframe in APL programming language for ease of experimentation, and then transported to a PC environment in BASIC for better portability. The recognition rate achieved was 94%, with a recognition speed of 10.6 characters/minute, running on a PC/AT with a math co-processor.

Pattern recognition Character recognition Cursive Arabic text recognition
Statistical techniques Segmentation Invariant moments Cutting models

1. INTRODUCTION

Character recognition is one of the applications of pattern recognition which is still a very challenging problem. Pattern recognition was always a very tempting area of research, especially if we consider that its ultimate goal is to simulate human vision. Similarly, the ultimate goal of character recognition is to simulate the human reading capabilities. Unlike pattern recognition, which is still not very successful, character recognition has achieved considerable progress.

Character recognition systems view the characters as two completely distinct research domains. The first is concerned with handwritten characters, and the second with typewritten characters. This distinction is due to the difference in complexity levels, as well as the possible practical usage of the two types of characters. Recognition systems for typewritten text are practically available for several Latin-based languages and mainly for English, with recognition rates reaching 99%. However, systems for handwritten text still impose several constraints on the writer to achieve acceptable recognition rates.

Automatic character recognition applications are numerous. Considering the recognition of typewritten text, we can envisage a variety of applications that can benefit from such systems. Of course, a practical system that is able to recognize handwritten text would be of much more help, but unfortunately it is not yet practically available. In the following paragraphs we will review some of the possible applications of typewritten text recognition.

Data entry is one of the applications that could benefit from such systems. The data could simply be typed on a normal typewriter and then scanned and recognized by such a system. This process could spare computer terminals for more important tasks; it also makes use of typewriters in the data entry task, which is cheaper than data entry equipment.

For archive and text retrieval applications, automatic character recognition could save invaluable time in data entry. Normally, the input of archiving and text retrieval systems is printed documents, such as incoming letters, documents, books, etc. Character recognition systems could reduce considerably the time between receiving the document and adding it to the data bases. The speed of such systems would be much greater and more economical than manual data entry. This fact will allow more practical means for creating text retrieval systems.

The scope of the recognition system addressed here (Fig. 1) is limited to the recognition of typewritten fonts. Moreover, we limited our system to a font-

* To whom correspondence should be addressed at IBM Bergen Scientific Center, Allegaten 36, 5007-Bergen, Norway.

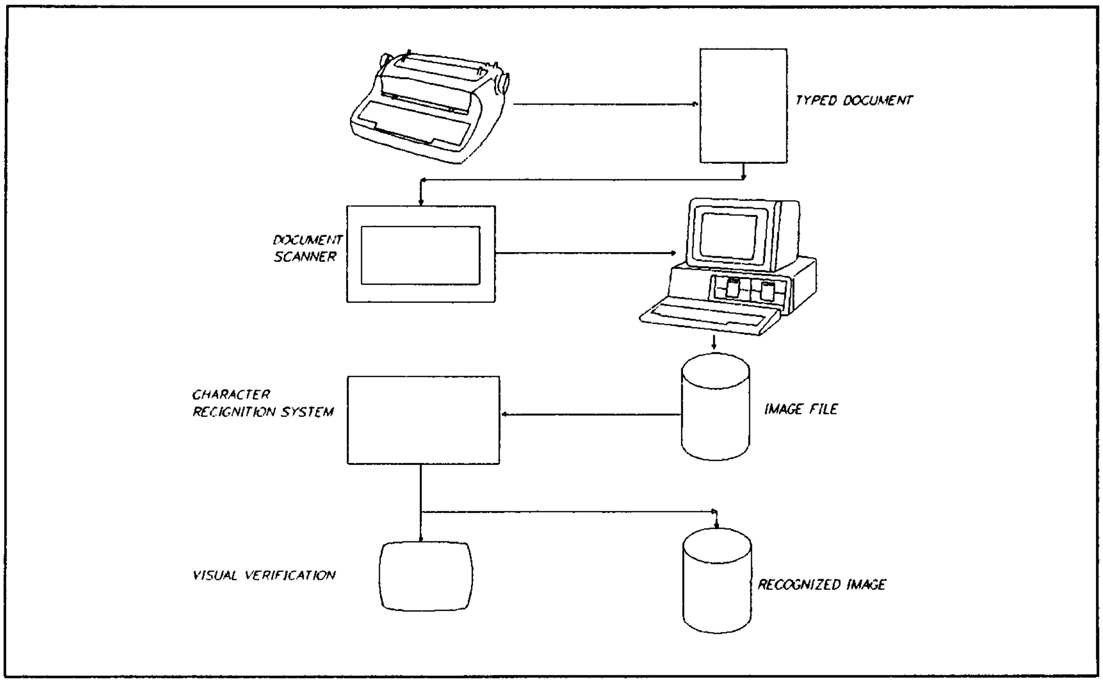


Fig. 1. Suggested character recognition system.

dependent recognition system, which improved recognition rate and speed.

2. ARABIC SCRIPT CHARACTERS

This section is meant for those who are not familiar with the characteristics of Arabic script. It describes the Arabic script and font characteristics that make it different from English, especially for the character recognition problem. These differences will show that Arabic character recognition is not a direct implementation of the recognition techniques used for English.

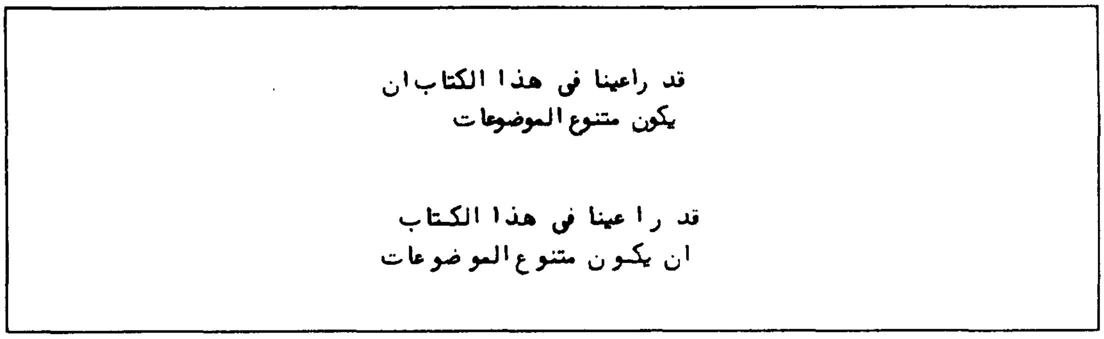
- Arabic text is written from right to left and recognition is expected to be in the same direction to allow for connection with a speech synthesis machine or linguistic checking of the recognized words.
- Whereas English characters can appear in two shapes (upper and lower case), Arabic characters can have four different shapes, depending on the position in the word (beginning, middle, end or alone). Unlike English, most of the Arabic characters of a word are connected along a base line.
- Many Arabic characters have dots which are positioned at a suitable distance above or below the letter body. Dots can be single, double, or triple. Different Arabic letters can have the same body and differ in the number of dots identifying them.
- Some Arabic characters use special marks to modify the letter accent, such as Hamza (ء), Mada (ٓ), again positioned at certain distances from the letter.
- Arabic uses another type of special character as short vowels, which are referred to as diacritics. Although different diacritics on the same characters could lead to different words, an Arabic reader is trained to deduce the meaning of undiacriticized text. This is why diacritics are not used in newspapers or in office correspondence. However, when diacritics are used they appear above or below the characters and are viewed as isolated characters.
- Arabic characters, as described by calligraphers, are composed of two parts; the crown, which identifies the character in any position (beginning, middle, end or alone), and the connection to neighboring characters. This feature proved to be very useful in the implementation of our character recognition system.
- To preserve the beauty of Arabic script, some letters are positioned to overlap with the neighboring letters as shown in Fig. 2. The degree of overlapping can vary according to the typeface and the typewriter design. For character recognition this could be a very important feature, as it is commonly used.

3. RECOGNITION APPROACH

It is possible to group the various techniques of character recognition under the two following headings:

- statistical or decision-theoretic approach;
- syntactic or linguistic approach.

In the statistical approach a set of characteristic measurements are extracted for each pattern. These characteristic measurements are called features of the



قد راعينا في هذا الكتاب ان
يكون متنوع الموضوعات

قد را عينا في هذا الكتاب
ان يكون متنوع الموضوعات

Fig. 2. Arabic text with and without overlapping.

pattern and they are used to recognize it. All the features of all the patterns, together with their pattern assignments, compose the feature space[1] of the recognition system. In a properly chosen feature space every pattern is assigned a unique partition.

In the syntactic approach[2] each pattern is expressed in terms of its components, called sub-patterns and pattern primitives. The recognition of each pattern is usually done by parsing the pattern structure according to a given set of syntax rules.

In the statistical approach recognition is based on the features extracted from the input pattern. These features are supposed to be invariant or less sensitive to distortions and variations. Although the patterns we are trying to recognize are typed characters, some variation is still expected as a result of improper alignment of text lines in the scanning process. Also, some variation is expected because of the typewriter quality.

In our implementation we used the invariant moments technique[3] to build the feature space. This measure is invariant with respect to size, translation and rotation. The calculations of the moments were implemented accumulatively in what we call Accumulative Moment Invariants. The accumulation feature served perfectly in the segmentation of Arabic characters.

3.1. *Mathematics of moment invariants*

The concept of moments is well established in classical mechanics. The two-dimensional $(p+q)$th order moments of a density distribution function $\rho(x,y)$ are defined in terms of a Riemann integral as

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q \rho(x,y)\,\mathrm{d}x\,\mathrm{d}y, \quad p,q = 0,1,2,\dots. \tag{1}$$

If it is assumed that $\rho(x,y)$ is a bounded function, and that it can have nonzero values only in the finite part of the xy plane, then moments of all orders exist and one can show that the double moment sequence $\{m_{pq}\}$ is uniquely determined by $\rho(x,y)$; and conversely, $\rho(x,y)$ is uniquely determined by $\{m_{pq}\}$.

The central moments μ_{pq} are defined as

$$\mu_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x-\bar{x})^p(y-\bar{y})^q\rho(x,y)\,\mathrm{d}(x-\bar{x})\,\mathrm{d}(y-\bar{y}), \tag{2}$$

where

$$\bar{x} = \frac{m_{10}}{m_{00}}, \quad \bar{y} = \frac{m_{01}}{m_{00}}. \tag{3}$$

It has been proved,[3] using the theory of algebraic invariants of binary algebraic forms, that out of the sequence $\{\mu_{pq}\}$ one generates a sequence $\{\hat{\mu}_r\}$ which is invariant under pattern translation, scaling and rotation. These are termed "invariant moments".

As a pattern is uniquely defined in terms of its sequence of moments and similarly of its sequence of invariant moments, one is then led to construct a pattern feature space whose features are the invariant moments. Moreover, the very definition of a moment in terms of a Riemann integral gives it an accumulative nature, a property which will be exploited further (see Section 3.2) for automatic character segmentation.

3.2. *Accumulative moment invariants*

For the distribution function $\rho(x,y)$ defined in the finite intervals $b \geq x \geq a$ and $d \geq y \geq c$, consider for example the invariant moment $\hat{\mu}_1$ defined by[3]

$$\hat{\mu}_1 = \mu_{20} + \mu_{02}. \tag{4}$$

Then for the rectangle $e_1 \geq x \geq a$, $d \geq y \geq c$, if one rewrites the part μ_{20}, as shown in equation (2), as

$$\mu_{20}(a,e_1) = \int_c^d\int_a^{e_1} x^2\rho(x,y)\,\mathrm{d}x\,\mathrm{d}y - \bar{x}^2\int_c^d\int_a^{e_1}\rho(x,y)\,\mathrm{d}x\,\mathrm{d}y, \tag{5}$$

with

$$\bar{x} = \frac{\int_c^d\int_a^{e_1} x\rho(x,y)\,\mathrm{d}x\,\mathrm{d}y}{\int_c^d\int_a^{e_1}\rho(x,y)\,\mathrm{d}x\,\mathrm{d}y}, \tag{6}$$

defining

$$l_{pq}(a, e_1) = \int_c^d \int_a^{e_1} x^p y^q \rho(x, y) \, dx \, dy, \tag{7}$$

then $\mu_{20}(a, e_1)$ could be expressed in terms of $l_{20}(a, e_1)$ as

$$\mu_{20}(a, e_1) = l_{20}(a, e_1) - \frac{l_{10}^2(a, e_1)}{l_{00}(a, e_1)}. \tag{8}$$

Now, as

$$l_{pq}(a, e_2) = l_{pq}(a, e_1) + l_{pq}(e_1, e_2), \quad \text{with } e_2 \geq e_1 \geq a. \tag{9}$$

One is led to the following algorithm to calculate invariant moments accumulatively (again explained by the μ_{20} case):

- While calculating the μ_{20} of $\rho(x, y)$ in the rectangle $d \geq y \geq c$ and $e_1 \geq x \geq a$, store the results for $l_{20}(a, e_1)$, $l_{10}(a, e_1)$ and $l_{00}(a, e_1)$.
- Expanding the rectangle of calculations in the x direction such that $e_2 \geq x \geq a$ with $e_2 > e_1$, then
  - calculate $l_{20}(e_1, e_2)$, $l_{10}(e_1, e_2)$ and $l_{00}(e_1, e_2)$,
    - obtain $l_{20}(a, e_2)$ by adding $[l_{20}(a, e_1) + l_{20}(e_1, e_2)]$, and similarly for l_{10} and l_{00}.
- Store the new $l_{20}(a, e_2)$, $l_{10}(a, e_2)$ and $l_{00}(a, e_2)$ in place of $l_{20}(a, e_1)$, $l_{10}(a, e_1)$ and $l_{00}(a, e_1)$.

4. RECOGNITION SYSTEM

One can view character recognition systems as composed of the following components, where each has a specific function in the recognition process:

(1) scanning component,
(2) segmentation component,
(3) recognition component,
(4) character segmentation component (in our approach),
(5) post-editing component.

These components are described in the following subsections.

4.1. *Scanning component*

The text is captured as an image with characters as the black parts and paper as the white ones. The average resolution of scanning is 200 × 200 pixel per inch. This resolution was found to be sufficient for the recognition of many Arabic typewriter fonts. A whole document (A4 page) could be scanned within a reasonable memory size. The scanned image is then passed to the next stage of recognition system. Depending on the limitations of the computer memory, two approaches could be adopted to pass the image. The first approach is to store the scanned image on external media and read the portions to be processed, which could slow the recognition process but free storage for other functions. The second approach would be to keep the whole image data in the memory to be recognized immediately, which saves input/output time but uses more memory space.

4.2. *Segmentation component*

Segmentation[4-6] is the process of isolating the image objects to be passed to the recognition phase. The segmentation process for the character recognition problem can be divided into three levels; line, word and character. The three levels of segmentation are described in the following paragraphs.

4.2.1. *Line segmentation.* This locates the boundaries of each line of text in the document image. Line boundaries can be determined by looking for the horizontal gaps in the image. These are identified by a full row of pixels with zero value. The identified rows are then checked top down to determine the top and bottom of each text line. This process should consider the dots above and below the characters, otherwise they will be isolated as separate lines, giving wrong results. In the case of diacriticized text the line segmentation should also consider the distance allowed between lines compared with the distance between diacritics and characters themselves (see Fig. 3).

4.2.2. *Word segmentation.* This locates the boundaries of each word in an isolated line. Word boundaries can be determined by looking for the vertical gaps in the segmented line, and checking them to identify the beginning and ending of words. Unfortunately, this process could also isolate portions of words or even a single character. These isolated objects will be referred to as word portions, as they could be part of words as shown in Fig. 4.

4.2.3. *Character segmentation.* As mentioned earlier, Arabic characters are not isolated and they can overlap. This is why the segmentation using vertical gap segmentation is not enough to isolate the characters (see Fig. 5).

One approach to isolating the characters is to identify the base line of the word portion and then identify the base line portions connecting neighboring characters. This approach proved inadequate for isolating overlapping characters.

The approach we use is to segment the characters after recognizing them. The details of character segmentation are described in Section 4.4.

4.3. *Recognition component*

As described in the previous section, the segmentation process was able to separate word portions. These word portions are then processed by the recognition system. The recognition system is based on calculating the invariant moments (see Section 6) for each character, and using these values to identify the character. As the characters are not yet separated, we assume that the first column of the word portion is a character. To save computation time, we do not use one column, but use the width of the smallest character in the set as a minimum width. The moments

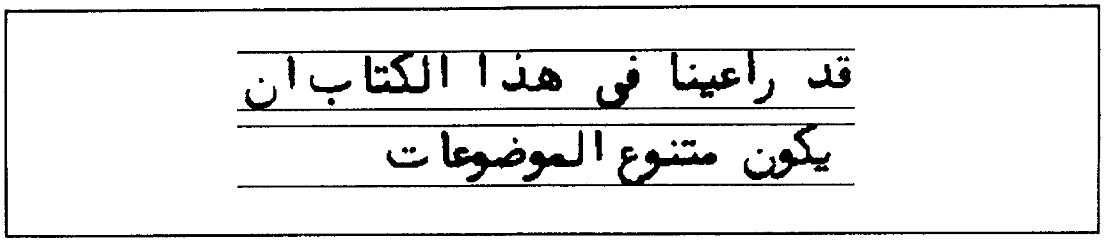


Fig. 3. Arabic script line segmentation.

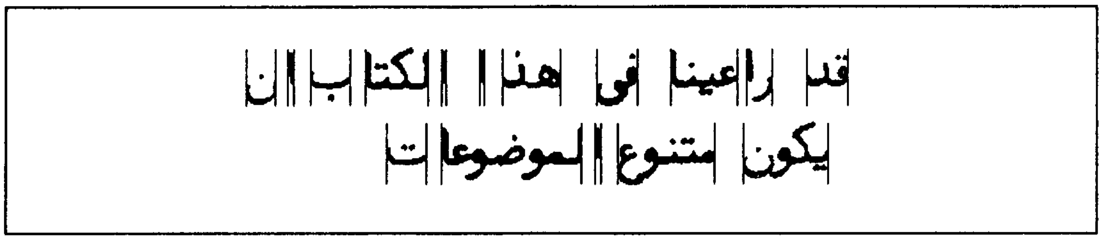


Fig. 4. Arabic script word segmentation.

are then calculated and checked against the feature space of that font. If the character (minimum width) is not found, another column is added to the character, then the moments are calculated again and checked against the feature space. This process is repeated until the character is recognized or the end of the word portion is reached. This process will be referred as column-by-column recognition. It was noted that for Arabic, the characters could be recognized before reaching their actual end. This feature allowed the system to handle overlapping and to isolate the connecting base line between connected characters.

In our implementation we have used the accumulative property of moments (described in Section 3.2) to avoid repetition in the calculations of the moments, which improved the recognition speed.

One of the drawbacks of column-by-column recognition is treating the word portion as the minimum input for recognition. If the recognition fails to recognize a letter in any part of the word portion, especially in the middle parts, the rest of the word portion will not be processed. To overcome that effect we introduced a backward recognition process, which starts processing the word portion from left to right. This process is applied only if the system failed to recognize a character in a word portion before its end, and ends where the forward recognition stopped.

4.4. *Character segmentation*

As mentioned earlier, the invariant moments calculated for the Arabic fonts are calculated to recognize characters before their actual end. Knowing that, we added to the feature space an extra feature describing the proper cutting edge to remove a segment of a character from the word portion. This cutting edge will be simply a vertical line in the case of non-overlapping characters. However, for overlapping characters the cutting edge will have different profiles according to the character recognized. This process is actually character segmentation, which we preferred to perform after recognition. The details of the cutting edges for the overlapping characters and character segmentation are given in the following sections.

The following Arabic characters were found to allow for overlapping: "Kaf" ـك, "Ra" ـر, "Za" ـز, and "Waw" ـو. It was also found that these characters could be grouped into two sets. The first set tends to overlap below the base line "Ra" ـر, "Za" ـز and "Waw" ـو; we call this descending overlap. The second set tends to overlap above the base line as in the case of "Kaf"ـك; we call this ascending overlap. According to the classification of the overlapping characters, two cutting models (segmentation) were used.

4.4.1. *Segmentation of descending overlap.* A model was designed to segment letters with descending overlap such as "Ra" ـر, "Za" ـز and "Waw" ـو from the following characters as shown in Fig. 6. It can be seen in the figure that to isolate the letter "Waw" ـو from the rest of the word, the letter "Waw" ـو has to be recognized before its end, and before the overlap. The model is then applied to remove the remaining part of the letter "Waw" ـو without affecting the following character.

Assuming that the recognition of the character was made at the column (c1), the model will try to find topmost non-blank pixel along this column. This point is the starting point for segmentation and we refer to it as (p1). Starting from (p1) the model will move horizontally in the recognition direction until it reaches a non-blank pixel, which will be part of the next character. A maximum distance is set for that direction where the model will stop, if it does not find non-blank pixel. We refer to this point as (p2) and

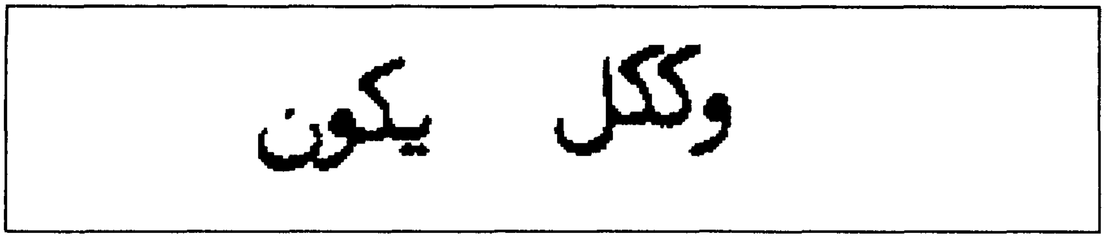


Fig. 5. Example of letters overlapping.

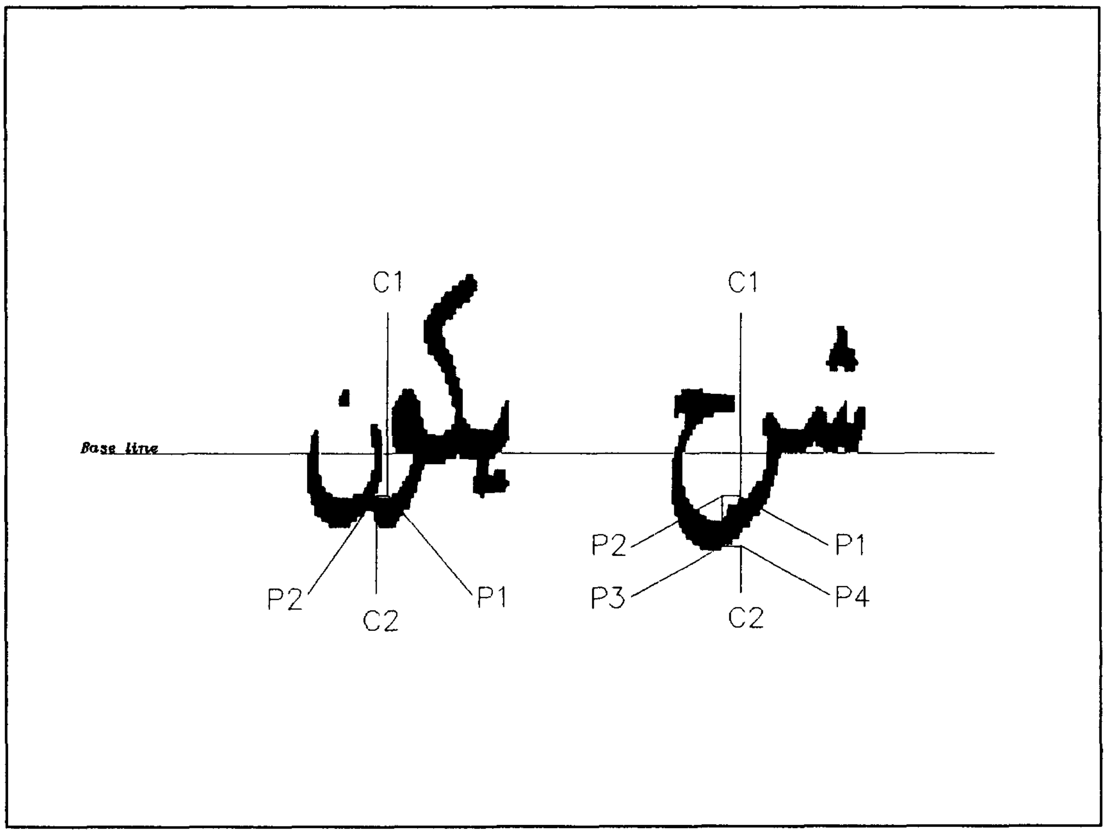


Fig. 6. Segmentation of descending overlap.

the columns at this point as (c2). This cutting edge represented by c1–p1–p2–c2 is normally enough for descending overlap except for sequences such as 'Aeen" and "Ra" رع. For such overlap sequences the model will apply an extra task to further specify the cutting edge. From point (p2) the model will move downwards along the column (c2) until it reaches a non-blank pixel or a maximum is reached. This point is referred to as (p3). The model will then move horizontally until (c1) and this point will be (p4). The cutting edge represented by c1–p1–p2–p3–p4–c2 was capable of segmenting all the descending overlap of the Arabic character set.

4.4.2. *Segmentation of ascending overlap.* Another model was designed for segmentation of the ascending overlap of the Arabic characters, such as the letter "Kaf" ك. This model is similar to that for the descending overlap but slightly different. In accordance with our assumption, the recognition of the character was made at column (c1), which is before the actual end of the character and before any overlap. The model moves from the top to bottom along (c1) until it finds a non-blank pixel, which will be (p1). From (p1) it moves horizontally in the recognition direction to (p2) at column (c2). From (p2) it moves downwards to (p3) on column (c2). From (p3) the model will move horizontally in a direction opposite to the recognition direction until it reaches a non-blank pixel at (p4) on column (c3). The cutting edge will be c1–p1–p2–p3–p4–c3. This cutting edge was found to be able to segment the difficult overlap of double "Kaf" كك. The ascending overlap and the cutting model are shown in Fig. 7.

4.5. *Post-editing component*

This component has two main functions: first, to allow the user to follow the recognition process, and

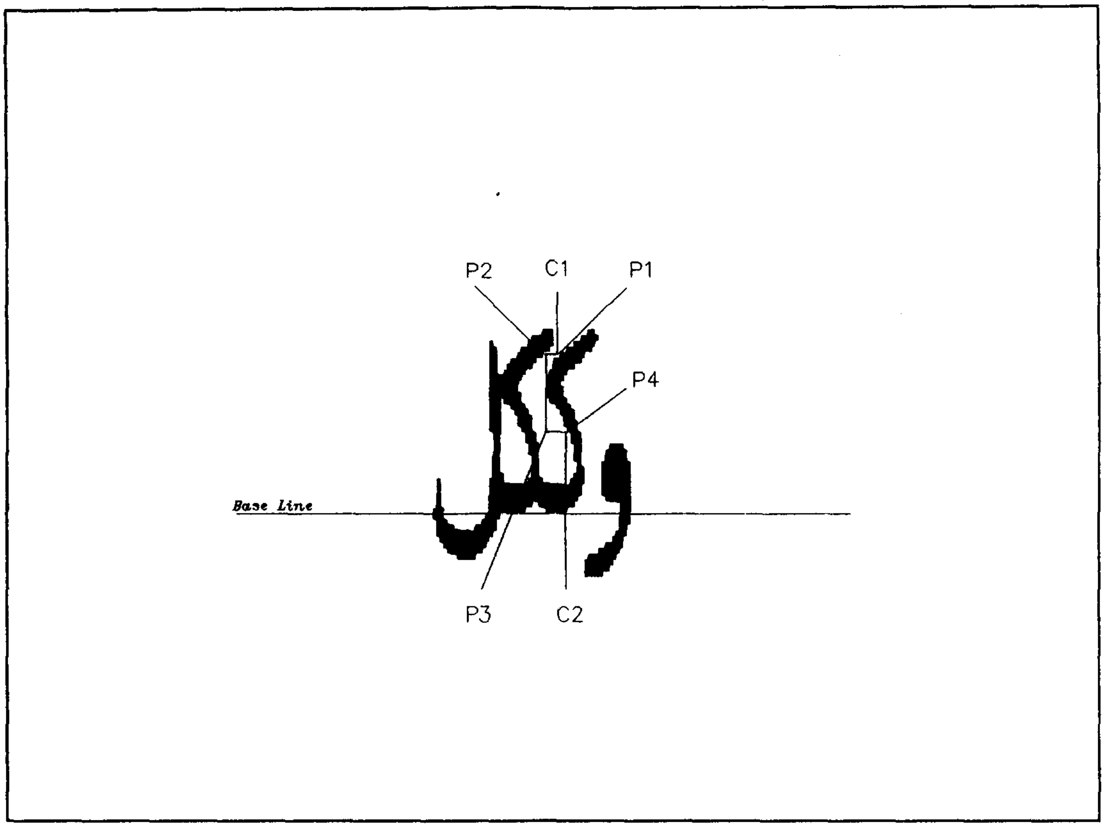


Fig. 7. Segmentation of ascending overlap.

second, to give the user a tool to correct the recognized text for mis-recognized or unrecognized characters.

A typical design of this tool would be a computer screen divided into two parts. The first part shows the graphic image of the input document with a pointer indicating the position where recognition is taking place. The second half shows the recognized text and continually adds recognized characters. When the recognition process is completed, the user is allowed to use the text part of the screen to edit the file and proof-read the document.

5. LEARNING PHASE AND FEATURE SPACE

The learning phase is an expression used to describe the process of making the system able to recognize a specific font. In other words, it is the process of building the feature space that will be used to recognize the individual characters of a font.

There are two approaches to build the feature space or train the system. The first is an interactive approach, where the user gives a character to the system and lets it calculate its features. The second approach is a batch system, where all the characters of a font are passed to the system to create the feature space for all the characters. Whereas the interactive approach is easier to use, we felt that the batch approach could be more practical, especially for users with limited computer experience.

The feature space in our case is composed of a number of moment invariants associated with specific characters of the font. These moments are kept in tables to be used during the recognition. The number of moments calculated is 7 plus 4 signs (skew moments to recognize the character orientation and identify it from mirror-image characters).

The process of moment calculation (feature space building) is done only once for each font that the system will recognize. Normally, these tables are created before the recognition process. The recognition system cannot depend entirely on the exact values of the moment invariants (a single point in the feature space). Therefore, the system allows for a specified tolerance (a zone in the feature space) for each moment invariant where the character will still be recognized. This is introduced to compensate for two major factors that influence the process of creating the feature space:

- *Scanner resolution effect*: The resolution of the scanner and the position of the character at scanning time affects the recognition process. If we consider a limited resolution scanner (200 × 200 pixel/inch), and small typewriter fonts (8 or 10 point), then the position of the character while scanning could result in one column of pixels being missed or added. This column affects the moment invariants, resulting in mis-recognition of the character.

- *Variation of typewriter shape*: It was noticed that some typewriters print slightly different shapes for the same letter. These shape differences could result from paper quality and strike variation even for electric typewriters. Although these differences could not be noticed by the normal reader, they could affect the resulting moment invariants considerably.

To overcome these two problems we scanned a reasonable number of samples of each letter (30 samples or more), and the moments were then calculated for each of them. For more details, see Section 6.

To calculate the zone in the feature space where each letter is identified, the mean and standard deviation are calculated for the moment invariants of each letter. The mean marks the center of the recognition zone, and the standard deviation identifies the size of the zone where the character is recognized. This zone will represent the allowable tolerance to compensate for the combined effect of the typewriter shape variation and scanner resolution effect.

6. IMPLEMENTATION ASPECTS

Extensive experiments were implemented using the moment technique as an approach for character recognition. The main goal of the testing is to prove whether the method is feasible or not, and if feasible, what are the criteria to achieve the best results. The experiments and the results are discussed in the following paragraphs.

6.1. *Weights and moment approach*

In an attempt to overcome the variation of typewriter letter shapes combined with the scanner resolution effect, we tried to assign different weights to the character pixels. The objective was to give less weight to the pixels on the contours and more weight to those in the body of the letter.

It could be concluded from equation (1) and the fact that our $\rho(x, y)$ takes only 1 and 0 values that the character pixels away from the center of gravity $(\bar{x}, \bar{y})$ have an influence on the moments calculation, which is larger than for pixels closer to the center of gravity. Therefore, to control this effect, one could modify the $\rho(x, y)$ definition by introducing a weighing factor $w(x, y)$ as follows:

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q \rho'(x, y)\,\mathrm{d}x\,\mathrm{d}y, \qquad (10)$$

where

$$\rho'(x, y) = w(x, y)\rho(x, y).$$

This $w(x, y)$, if used effectively, could help to model and compensate for the effects of typewriter and scanner on the input patterns of the recognition system (input text).

Different weighing functions were used and the results were compared with unweighted results. After several attempts we were convinced that introducing weights always reduced the recognition efficiency.

6.2. *Number of moments*

We have used the following six absolute moment invariants:

$$\hat{\mu}_1 = \mu_{20} - \mu_{02} \qquad (11)$$

$$\hat{\mu}_2 = \{(\mu_{20} + \mu_{02})^2 + 4\mu_{11}^2\}^{1/2} \qquad (12)$$

$$\hat{\mu}_3 = \{(\mu_{30} - \mu_{12})^2 + (3\mu_{21} - \mu_{03})^2\}^{1/2} \qquad (13)$$

$$\hat{\mu}_4 = \{(\mu_{30} - 3\mu_{12})^2 + (\mu_{21} + \mu_{03})^2\}^{1/2} \qquad (14)$$

$$\begin{aligned}\hat{\mu}_5 = \{&(\mu_{30} - 3\mu_{12})(\mu_{30} + \mu_{21})\\ &[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2]\\ &+ (3\mu_{21} - \mu_{03})(\mu_{21} + \mu_{03})\\ &[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2]\}^{1/4}\end{aligned} \qquad (15)$$

$$\begin{aligned}\hat{\mu}_6 = \{&(\mu_{20} - \mu_{02})[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2]\\ &+ 4\mu_{11}(\mu_{30} + \mu_{12})(\mu_{21} + \mu_{03})\}^{1/3},\end{aligned} \qquad (16)$$

and one skew moment invariant:

$$\begin{aligned}\dot{\mu}_7 = &(3\mu_{21} - \mu_{03})(\mu_{30} + \mu_{12})\\ &[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2]\\ &- (\mu_{30} - 3\mu_{12})(\mu_{21} + \mu_{03})\\ &[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2]\end{aligned} \qquad (17)$$

$$\hat{\mu}_7 = \{|\dot{\mu}_7|\}^{1/4}. \qquad (17.1)$$

The number of moments used is critical; a smaller number of moments will increase the recognition speed, whereas a larger number would improve recognition rate. The initial experiments were done on the 120 character shapes of Arabic, using the first two moments [equations (11) and (12)] as implemented in Ref. (3) for English. Plotting the calculated moments (Fig. 8), the results were not encouraging. Many characters were clustered near each other in the feature space, leaving no chance to recognize them uniquely. To improve recognition, a third moment invariant was added [equation (13)]. This extra moment represented a third dimension in the feature space and helped to identify the characters uniquely.

Now, considering the extra letter shapes (letter portions), resulting from the column-by-column calculation, the number of patterns processed by the recognition system increased drastically. Although the extra shapes of letter portions are not represented in the feature space we must consider them as part of the input patterns to be rejected. Accordingly, we had to increase the recognition constraints by adding extra moments (dimensions to the feature space) to prevent the letter portions from being recognized incorrectly as Arabic letters.

6.3. *Character orientation*

As mentioned earlier, the moment values are calculated to be invariant with respect to size and rotation.

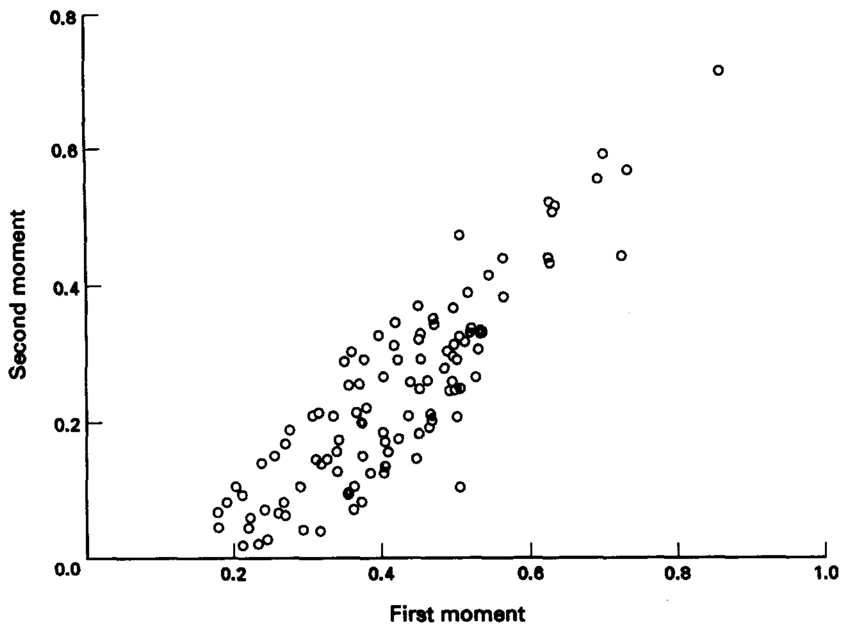


Fig. 8. Character distribution in the first and second moments feature space. The dark circles are characters with moment values almost the same. The adjacent circles demonstrate how close the moments are for different characters.

This characteristic helped to solve the variation due to slight rotations. Combining this characteristic with column-by-column recognition resulted in recognition of some letter portions as other letters, such as ـطـ and عـ. To overcome this problem the sign of skew moment was introduced to the feature space, whereas Hu(3) used only one skew moment in his implementation [equation (17)], we had to use the signs of four skew moments for Arabic. The first, for letters with mirror-image shapes along the vertical axis is the sign of equation (17) as given by equation (19). The second is the sign of equation (18) as given in equation (20), for letters with mirror-image shapes along the horizontal axis, as in the letter ـه and مـ. The other two signs are calculated to detect the 90° and 180° rotations as shown in equations (21) and (22) respectively.

$$\begin{aligned}\dot{\mu}_8 = {} & (3\mu_{12} - \mu_{30})(\mu_{03} + \mu_{21}) \\ & [(\mu_{03} + \mu_{21})^2 - 3(\mu_{12} + \mu_{30})^2] \\ & - (\mu_{03} - 3\mu_{21})(\mu_{12} + \mu_{30}) \\ & [3(\mu_{03} + \mu_{21})^2 - (\mu_{12} + \mu_{30})^2]\end{aligned} \quad (18)$$

$$s_1 = \text{Sign}\{\dot{\mu}_7\} \quad (19)$$

$$s_2 = \text{Sign}\{\dot{\mu}_8\} \quad (20)$$

$$s_3 = \text{Sign}\left\{\frac{\mu_{30}}{\mu_{00}^{2.5}}\right\} \quad (21)$$

$$s_4 = \text{Sign}\left\{\frac{\mu_{03}}{\mu_{00}^{2.5}}\right\}. \quad (22)$$

6.4. *Recognition-zone shape*

As mentioned in Section 5, we had to overcome the variations resulting from both scanning resolution and typewriter quality. To overcome this problem, we built the feature space by typing and scanning reasonable samples (30 or more) of the letter. These samples include the combined effects of typewriter and scanner, and, using these samples, we tried to identify the zone in the feature space where the letter will be recognized.

The center of the recognition zone was calculated to be the mean of the samples for each moment. To identify the zone the standard deviation was calculated for each moment. Using the mean and standard deviation, we implemented several techniques to identify the recognition zone. We experimented using a hyper box, hyper sphere and hyper ellipsoid. The best recognition was obtained using a hyper box, where the mean represents the center along each dimension, and the mean plus and minus the standard deviation of each dimension represents the boundaries of the box.

With the previous set-up the recognition rate was reasonably good, yet some valid characters were not recognized at all. That led us to introduce an empirical multiplier to the standard deviation to enlarge the recognition zone. By doing this for each character we increased the chance of zone overlapping in the feature space, which we overcame by using extra moments. We finally used a total of seven moments to reach a practical recognition rate. The empirical multiplier was also used as a tuning factor for some

Table 1. Performance vs number of moments

No. of moments used	Recognition rate (%)	Recognition speed (character/minute)
11	94	10.6
10	82	8.8
09	82	10.5
08	76	8.6
07	76	9.7
06	50	11.1
05	Below 50	

letters to increase or decrease their recognition zones selectively.

6.5. *Recognition positioning*

At one point in our work we considered representing some shapes of the same letter as one letter in the feature space. Unfortunately, it was found that small differences, as in letter "Ba" at the beginning (بـ) and at the middle (ـبـ), still affect recognition. In our final implementation we represented all the four shapes of a letter in our feature space.

As the four shapes of letters were considered, it is necessary to start the recognition process at the right position for the inter-word letters or the letter will not be recognized. It is not possible to determine the exact position of the next letter, so the recognition process starts after the isolation is done and after skipping any elongation on the base line. If the process fails to recognize the letter, it will automatically reposition itself to the left and/or to the right, and attempt to recognize the letter. If all the trials for repositioning are exhausted, and the letter is still not recognized, the back-scanning recognition process is invoked.

6.6. *Performance*

To measure the performance of the system, a PC/AT equipped with a math co-processor was used to run it. The sample test was four words with a total of 17 characters. As shown in the table, the test was carried out using different numbers of moments. The recognition rate showed good improvement by increasing the number of moments to 11 moments. However, using fewer moments does not necessarily improve the recognition speed, as a result of the invocation of the recognition positioning and backward recognition process and the excessive search for a match.

7. CONCLUSION

The solution presented in this paper for the problem of typewritten Arabic character recognition using the statistical method proved to be very practical in solving some of the typical problems of Arabic script. By overcoming the overlap problem of Arabic characters, the system managed to handle a larger number of typewritten fonts.

Although we used only a small sample for performance evaluation, the system is capable of handling large samples of data involving several pages of text. The PC implementation imposes no memory limitation, as the system can scan several pages and store them in external memory (disk). The pages, or parts of them, are then recalled to the memory for processing, which we call batch mode. For a single page, or portions of a page, the text is scanned directly to the memory, saving disk read write time, which we call realtime mode. An A4 page would consume approximately half a megabyte of storage, which is within the capacity of current PCs.

The recognition speed on PC/AT is still slow for realtime mode. However, the performance (speed) could be improved drastically if a more powerful system, such as PS/2 model 80, is used. The speed of the current implementation is acceptable for batch mode.

Some of the limitations of the system were the result of the heavy computation involved in the moments approach and its sensitivity to the smallest variation in the input patterns. Using a large sample helped to identify the recognition zone to a great extent, but it increased the possibility of mis-recognition. Another limitation was because in the column-by-column recognition some parts of word portions were not passed to the recognition component when the system failed to recognize an earlier part. This problem was partly solved by using backward recognition.

However, we feel that there is room for improvement and there are possibilities of using a syntactic approach to reduce the iterations and calculations.

REFERENCES

1. K. S. Fu, *Sequential Methods in Pattern Recognition and Machine Learning*. Academic Press, New York (1968).
2. H. Almuallim and S. Yamaguchi, A method of recognition of Arabic-cursive handwriting, *IEEE Trans. Pattern Anal. Machine Intell.* **PAMI-8**(5), 715–722 (1987).
3. Ming-kuei Hu, Visual pattern recognition by moment invariants, *IRE Trans. Inform. Theory* **IT-8**, 179–187 (1962).
4. J. R. Ullmann, Advances in character recognition, *Application of Pattern Recognition*, K. S. Fu, Ed. CRC Press, FL (1962).
5. A. Amin, Arabic handwriting recognition and understanding, *Computer processing of the Arabic Language Workshop*, April 1985, Kuwait.
6. N. Ayache and O. D. Faugeras, Hyper: a new approach for the recognition and positioning of two-dimensional objects, *IEEE Trans. Pattern Anal. Machine Intell.* **PAMI-8**(1), 44–54 (1986).

About the Author—SHERIF SAMI EL-DABI received the B.Sc. degree in Mechanical Engineering from Cairo University (1971). He is currently the manager of IBM Kuwait Scientific Center. His research interests include Arabization of text processing and retrieval systems, computer-aided font design for Arabic, and Arabic computational linguistics.

About the Author—REFAT RAMSIS received the B.Sc. degree in Computer Science and Mathematical Statistics from Kuwait University (1980). He has worked as a systems programmer; currently he is database designer at the National Computer and Microfilm Center (NCMC). His research interests include computer-aided font design for Arabic.

About the Author—ALADIN KAMEL received the B.Sc. (Hons) degree in Electrical Engineering from Ain Shams University, Cairo (1975), the B.Sc. (Hons) degree in Pure and Applied Mathematics from Ain Shams University, Cairo (1978) and the Ph.D. degree in Electrical Engineering from the Polytechnic University, New York (1981). He is currently a senior scientist with Bergen Scientific Center, IBM Norway. His research interests include digital image processing and parallel/vector processing.